UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/761,122	01/16/2001	Shinichi Yamaura	2271/64016	2922

7590 01/20/2006

Ivan S. Kavrukov
Cooper & Dunham LLP
1185 Avenue of the Americas
New York, NY 10036

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 01/20/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	09/761,122	YAMAURA ET AL.
	Examiner	Art Unit
	David J. Huisman	2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *03 November 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11 and 23* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11 and 23* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *02 November 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☒ All b)☐ Some * c)☐ None of:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-11 and 23 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 11/3/2005.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-2, 5-7, 10-11, and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Li et al., U.S. Patent No. 4,783,738 (as applied in the previous Office Action and herein referred to as Li), in view of Hennessy and Patterson, "Computer Architecture - A Quantitative Approach, 2nd Edition," 1996 (herein referred to as Hennessy).

5. Referring to claim 1, Li has taught a parallel processor comprising:

a) a global processor interpreting a program and controlling the entirety of the parallel processor. See Fig.1, component 2, and column 5, lines 59-61.

b) a processor-element block (Fig.1, component 1) comprising a plurality of processor elements (Fig.1, components 5) each comprising a register file and an operation array for processing a plurality of sets of data. See column 7, lines 15-17, and note that each element multiple registers

(register file), and looking at Fig.1, it can be seen that each element includes an operation array (components 72 (ALU) and 73) for operating on data.

c) wherein said global processor assigns to said plurality of processor elements respective processor-element numbers and outputs a control signal to said plurality of processor elements, and, thereby, sets the assigned processor-element numbers corresponding to said plurality of processor elements as input values of the operation arrays, respectively. See column 5, lines 44-46, and also see Fig.2.

d) processing by the operation array of the processor element, according to processor element instructions from the global processor. See column 5, lines 59-60, and note that instructions are passed from global processor (Fig.1, component 2) to processing elements (Fig.1, components 5) across a communication bus (Fig.1, component 3). One example is given in column 10, lines 49-50, where it is stated that an ADD/SUB instruction may be sent to processing elements.

e) Although Li has taught registers (column 7, lines 15-17) and that an ALU 72 (Fig.1) in a processing element receives operands from a local memory 71 (Fig.1), Li has not explicitly taught that for each processor element said register file of the processor element holds plural sets of data for operation. That is, Li has not disclosed that the local memory of Fig.1 is absolutely a register file. However, Hennessy has taught that register files are local memories which store and supply data operands in a processor. See pages 40 (Fig.1.15), 41 (Fig.1.16), and 130. Register files are the fastest form of memory and therefore, they may provide operands at a very high speed. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li such that Li's local memory (in each processing element) is a register file. This would ensure that the ALU is able to retrieve operands as quickly as possible.

f) Li has further taught that the operation array of the processor element includes a multiplexer coupled to the register file of the processor element and to the register files of respective adjacent processor elements. Recall that the operation array includes components 72 and 73 in Fig.1. Component 73 includes DSV 78. DSV 78 in turn includes a multiplexer. See Fig.2-5. This multiplexer is coupled to local memory (register file) 71 through the ALU of the processor element shown in Fig.1. In addition, as shown in Fig.1, the processor elements are coupled to one another, thereby forming an array of processor elements (components 5). Consequently, the multiplexer of one processor element is coupled to the register files of adjacent elements since the one processor element is coupled to the adjacent elements.

6. Referring to claim 2, Li in view of Hennessy has taught a parallel processor as described in claim 1. Li has further taught that data is transferred from the global processor to the coprocessors. See column 5, lines 59-61, and Fig.1. Li has not explicitly taught that the transferred data came from the register file of the global processor. However, Official Notice is taken that register files and register file transfers between multiple processors are well known and accepted in the art. Data is stored in memories such as register files, data caches, main memory, hard disk, etc. However, register files are accessed the quickest, and therefore, would result in the fastest data transfer. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to transfer data from the register file of Li's global processor to the processing elements.

7. Referring to claim 5, Li in view of Hennessy has taught a processor as described in claim 2. Li has not taught that the data transfer is rendered through specifying by a pointer using a general-purpose register of said global processor. However, Hennessy has taught specifying a

pointer using a general-purpose register on page 75 (see the indirect addressing mode). This mode is also used when loading (transferring) data to a register. See page 100. By implementing this indirect mode, Li's system would not be restricted to merely transferring data from register file to register file. Instead, by using pointers, data can be transferred from main memory, which would be beneficial just in case a lot of data needs to be transferred and there aren't enough registers to store the data. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li in view of Hennessy such that a data transfer is rendered by specifying a pointer using a register.

8. Referring to claim 6, Li in view of Hennessy has taught a processor as described in claim 5. Li has not taught that the specifying by a pointer comprises incrementing of data in said general-purpose register after the specifying. However, Hennessy has taught incrementing the contents of the register after the specifying. See the autoincrement mode on page 75. As explained in Fig.2.5 of Hennessy, the autoincrement mode is useful for transferring a block of data, for instance, which may be governed by a loop. By implementing this mode, Li's system would not be restricted to merely transferring data from register file to register file. Instead, by using pointers, data can be transferred from main memory, which would be beneficial just in case a lot of data needs to be transferred and there aren't enough registers to store the data. In addition, if a block of data needs to be transferred, a second instruction would not be needed to increment the register data each time because the single instruction does it automatically. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li in view of Hennessy such that after specifying a pointer using a register, incrementing the contents of the register.

9. Referring to claim 7, Li in view of Hennessy has taught a parallel processor as described in claim 1. Li has further taught that each processor element comprises a plurality of flag bits for controlling, according to a state of the data, as to whether or not the operation processing is to be executed, according to whether or not a condition is satisfied, and, renders AND/OR operation on a specific bit of said flag bits. See column 10, and note that a PE will execute if its agreement bit is true. The agreement bit is determined by at least AND/OR operations on different flags. See column 8, lines 3-7.

10. Referring to claim 10, Li in view of Hennessy has taught a parallel processor as described in claim 7. Li has not explicitly taught that specifying of said flag bits is rendered through specifying by a pointer using a general-purpose register of said global processor. However, Hennessy has taught specifying data via specifying a pointer using a general-purpose register on page 75 (see the indirect addressing mode). By implementing this indirect mode, Li's system would not be restricted to using immediate values to specify flag bits. Instead, by using pointers, the flag bits may be produced from main memory, thereby offering greater flexibility to the programmer. This may be beneficial because immediate values are not dependent on past operation. On the other hand, the programmer may want to use a processing-element configuration based on some data produced during previous execution. With indirect addressing, this data may be used to determine the appropriate flag bits, which specify the configuration of elements (which processing elements should be enabled), and the bits can be stored in main memory for future use. This flexibility is not available with immediate values only. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to

modify Li in view of Hennessy such that flag bits are specified via specifying a pointer using a register.

11. Referring to claim 11, Li in view of Hennessy has taught a parallel processor as described in claim 10. Li has not explicitly taught that the specifying by a pointer comprises incrementing of data in said general-purpose register after the specifying. However, Hennessy has taught incrementing the contents of the register after the specifying. See the autoincrement mode on page 75. As explained in Fig.2.5 of Hennessy, the autoincrement mode is useful for accessing linear data within a loop. By implementing this mode, Li's system would be able to quickly achieve different preset processing-element configurations (for instance, multiple combinations of flag bits may be stored in main memory). Instead of having two instructions to first access flag bits from one address and then updating the address itself, only one instruction is needed to both access the flag bits and update the address so a next address can be read from. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li in view of Hennessy such that after specifying a pointer using a register, incrementing the contents of the register.

12. Referring to claim 23, Li in view of Hennessy has taught a parallel processor as described in claim 1. Li has further taught that each processor element further includes a shift and expansion circuit. See Fig.2, component 82 and note that shifting occurs. In addition, element expansion occurs. More specifically, the system may choose to operate as a SIMD processor in which multiple elements operate on an instruction at a given time (see column 10, lines 31-53). So the number of elements working at once is expanded. The circuitry which allows a PE to become part of the current overall processing element is the expansion circuitry.

13. Claims 1-11 and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Kaneko et al., U.S. Patent No. 5,430,885 (as applied in the previous Office Action and herein referred to as Kaneko) in view of Hennessy, as applied above, and further in view of Miyata, U.S. Patent No. 4,858,110. Furthermore, Li, as applied above, is cited as extrinsic evidence for showing, in addition to Kaneko, that a global processor actually performs the assigning of processor numbers, even if they simply represent coordinates.

14. Referring to claim 1, Kaneko has taught a parallel processor comprising:

a) a global processor interpreting a program and controlling the entirety of the processor. See Fig.1, component 401.

b) processor-element block comprising a plurality of processor elements each comprising an operation array for processing a plurality of sets of data. See Fig.1, components 411, 412, 413, etc. Also, see column 1, lines 19-25. Note that coprocessors carry out tasks and therefore, would inherently contain an operation array for execution purposes. Although it is not explicitly disclosed that each coprocessor (assisting processor) contains a register file, Hennessy has taught that register files are common components which store and supply data operands in a processor. See pages 40, 41, and 130. Register files are the fastest form of memory and therefore, they may provide operands at a very high speed. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaneko such that each coprocessor contains a register file.

c) wherein said global processor assigns to said plurality of processor elements respective processor-element numbers, and outputs a control signal to said plurality of processor elements,

and, thereby, sets the assigned processor-element numbers corresponding to said plurality of processor elements as input values of the operation arrays, respectively. See column 3, lines 33-47, and note that each processor element (coprocessor) has a register (Fig.6, components 701 and 702, for instance) for storing a processing element number assigned to it by the host processor. And, from Fig.6, this processor number is inputted into at least a portion of the operation array (logic which processes data). Applicant's attention is also directed to Li, Fig.2, component 81, column 5, lines 44-46, column 7, lines 40-45, and column 9, lines 57-61. These passages show that a processor may do the assigning just as Kaneko does by saying the host processor "presents" processor numbers (column 3, lines 33-37).

d) wherein for each processor element said register file of the processor element holds plural sets of data for operation processing by the operation array of the processor element, according to processor element instructions from the global processor. See column 2, line 56, to column 3, line 15, and note that instructions are passed from the global processor to processing elements. Furthermore, as explained above, it would have been obvious for the coprocessors in Kaneko to have register files for holding data to be operated upon.

e) Kaneko in view of Hennessy has not taught that the operation array of the processor element includes a multiplexer coupled to the register file of the processor element and to the register files of respective adjacent processor elements. However, Miyata has taught such a concept. See Fig.8 and column 1, lines 32-46. As described such a connection allows for the transfer of data between register files of adjacent PEs. Since data-transferring PEs are coupled in Kaneko, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaneko in view of Hennessy to transfer data using a multiplexer as described in Miyata.

15. Referring to claim 2, Kaneko in view of Hennessy has taught a parallel processor as described in claim 1. Kaneko has further taught that data is transferred from the global processor to the coprocessors. See the abstract and column 2, lines 3-12. Kaneko has not explicitly taught that the transferred data came from the register file of the global processor. However, Official Notice is taken that register files and register file transfers between multiple processors are well known and accepted in the art. Data is generally stored in memories such as register files, data caches, main memory, hard disk, etc. However, register files are accessed the quickest, and therefore, would result in the fastest data transfer. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to transfer data from the register file of Kaneko's global processor to the processing elements.

16. Referring to claim 3, Kaneko in view of Hennessy has taught a parallel processor as described in claim 2. Kaneko has further taught that the data transfer is rendered through specifying a range from a first specific processor element through a second specific processor element by specifying immediate values by using operands. See Fig.12B and column 10, lines 14-29, and note that the global processor specifies an entire range of coprocessors by issuing a designation number in the form of P(a, b, c, d). Clearly, this will be an operand of some form of transmit (or communicate) instruction. Also, a, b, c, and d are integers, which are in turn, immediate values.

17. Referring to claim 4, Kaneko in view of Hennessy has taught a parallel processor as described in claim 2. Kaneko has further taught that the data transfer is rendered through specifying bits such as to specify processor elements matching processor-element numbers expressed in binary notation and specifying processor elements by masking arbitrary bits of the

thus-specified bits, through specifying the immediate values by using the operands. See Fig.6 and note that a comparison occurs between the number assigned to the processor (mask) and the number specified by the global processor. This assigned number is a mask because it is used to determine the value of bit locations in another value (the value specified by the global processor), which is the purpose of a mask. More specifically, the mask is used in a comparison. If the values are equal, then the system knows that the corresponding coprocessor will be used in communication. And, these numbers are integers, which are in turn immediate values.

18. Referring to claim 5, Kaneko in view of Hennessy has taught a parallel processor as described in claim 2. Kaneko has not taught that the data transfer is rendered through specifying by a pointer using a general-purpose register of said global processor. However, Hennessy has taught specifying a pointer using a general-purpose register on page 75 (see the indirect addressing mode). This mode is also used when loading (transferring) data to a register. See page 100. By implementing this indirect mode, Kaneko's system would not be restricted to merely transferring data from register file to register file. Instead, by using pointers, data can be transferred from main memory, which would be beneficial just in case a lot of data needs to be transferred and there aren't enough registers to store the data. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaneko in view of Hennessy such that a data transfer is rendered by specifying a pointer using a register.

19. Referring to claim 6, Kaneko in view of Hennessy has taught a parallel processor as described in claim 5. Kaneko has not taught that the specifying by a pointer comprises incrementing of data in said general-purpose register after the specifying. However, Hennessy has taught incrementing the contents of the register after the specifying. See the autoincrement

mode on page 75. As explained in Fig.2.5 of Hennessy, the autoincrement mode is useful for transferring a block of data, for instance, which may be governed by a loop. By implementing this mode, Kaneko's system would not be restricted to merely transferring data from register file to register file. Instead, by using pointers, data can be transferred from main memory, which would be beneficial just in case a lot of data needs to be transferred and there aren't enough registers to store the data. In addition, if a block of data needs to be transferred, a second instruction would not be needed to increment the register data each time because the single instruction does it automatically. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaneko in view of Hennessy such that after specifying a pointer using a register, incrementing the contents of the register.

20. Referring to claim 7, Kaneko in view of Hennessy has taught a parallel processor as described in claim 1. Kaneko has further taught that each processor element comprises a plurality of flag bits for controlling, according to a state of the data, as to whether or not the operation processing is to be executed, according to whether or not a condition is satisfied, and, renders AND/OR operation on a specific bit of said flag bits. Looking at Fig.8A, each processing element comprises (or is associated with) two flag bits; an X-coordinate flag bit 502 and a Y-coordinate flag bit 503. The global processor will specify X and Y vectors and if both the X and Y coordinate flag bits for each processing element are 1, then that processor will be enabled for execution purposes. So, for instance, looking at Fig.8A, processor 304 is enabled because its X-bit is set to 1 and its Y-bit is set to 1. However, the processing element directly left of element 304 is disabled because its X-bit is set to 0 and its Y-bit is set to 1. Therefore, it can be seen that an AND operation must be performed on the X-bit. That is, the X-bit AND the

Y-bit must be equal to 1 (XY = 1) for enablement to occur. See Fig.13 for an enablement flowchart.

21. Referring to claim 8, Kaneko in view of Hennessy has taught a parallel processor as described in claim 7. Kaneko has further taught that specifying of said flag bits is rendered by specifying the range from the first specific processor element through the second specific processor element through specifying the immediate values by using the operands. Again, see Fig.8A, and note that immediate values 01100 (502) and 0110 (503), which are specified by the global processor, correspond to a range of processors that will be enables. These immediate values contain the flag bits which specify the range of elements which will be enabled.

22. Referring to claim 9, Kaneko in view of Hennessy has taught a parallel processor as described in claim 7. Kaneko has further taught that specifying of said flag bits is rendered through specifying bits such as to specify processor elements matching processor-element numbers expressed in binary notation and specifying processor elements by masking arbitrary bits of the thus-specified bits through specifying the immediate values by using the operands. It should be realized that the flag bits merely play a part in determining which processing element(s) will be enabled. Furthermore, see Fig.6 and note that a comparison occurs between the number assigned to the processor (mask) and the number specified by the global processor. This assigned number is a mask because it is used to determine the value of bit locations in another value (the value specified by the global processor), which is the purpose of a mask. More specifically, the mask is used in a comparison. If the values are equal, then the system knows that the corresponding coprocessor will be enabled. And, these numbers are integers, which are in turn immediate values. Finally, even the output of AND gate 709 in Fig.6 is a flag

bit in that through specifying bits which either match or do not match the processor number, this flag is produced which either enables or disables the coprocessor from receiving input through input unit 710 for execution.

23. Referring to claim 10, Kaneko in view of Hennessy has taught a parallel processor as described in claim 7. Kaneko has not explicitly taught that specifying of said flag bits is rendered through specifying by a pointer using a general-purpose register of said global processor. However, Hennessy has taught specifying data via specifying a pointer using a general-purpose register on page 75 (see the indirect addressing mode). By implementing this indirect mode, Kaneko's system would not be restricted to using immediate values to specify flag bits. Instead, by using pointers, the flag bits may be produced from main memory, thereby offering greater flexibility to the programmer. This may be beneficial because immediate values are not dependent on past operation. On the other hand, the programmer may want to use a processing-element configuration based on some data produced during previous execution. With indirect addressing, this data may be used to determine the appropriate flag bits, which specify the configuration of elements (which processing elements should be enabled), and the bits can be stored in main memory for future use. This flexibility is not available with immediate values only. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaneko in view of Hennessy such that flag bits are specified via specifying a pointer using a register.

24. Referring to claim 11, Kaneko in view of Hennessy has taught a parallel processor as described in claim 10. Kaneko has not explicitly taught that the specifying by a pointer comprises incrementing of data in said general-purpose register after the specifying. However,

Hennessy has taught incrementing the contents of the register after the specifying. See the autoincrement mode on page 75. As explained in Fig.2.5 of Hennessy, the autoincrement mode is useful for accessing linear data within a loop. By implementing this mode, Kaneko's system would be able to quickly achieve different preset processing-element configurations (for instance, multiple combinations of flag bits may be stored in main memory). Instead of having two instructions to first access flag bits from one address and then updating the address itself, only one instruction is needed to both access the flag bits and update the address so a next address can be read from. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Kaneko in view of Hennessy such that after specifying a pointer using a register, incrementing the contents of the register.

25. Referring to claim 23, Kaneko in view of Hennessy and further in view of Miyata has taught a parallel processor as described in claim 1. Kaneko has further taught that each processor element further includes a shift and expansion circuit. See Fig.3A-D and Fig.14A. It should be realized that the number of processing elements operating at any given time may be expanded. For instance, at time T, four elements may be operating at once, as shown in Fig.3B. However, at some time T+X, every element may operate at once, as shown in Fig.3A. That is, the number of active elements has expanded. Each element is responsive to signals in order to make it active. The activating circuitry is the expansion circuitry. In addition, as shown in Fig.14A, coprocessors may transfer (shift) data to one another (i.e., data is shifted from one element to another). The I/O circuitry is the shift circuitry.

26. Claims 3-4 and 8-9 are rejected under 35 U.S.C. 103(a) as being unpatentable over Li in view of Hennessy, as applied above, in view of Kaneko, as applied above.

27. Referring to claim 3, Li in view of Hennessy has taught a parallel processor as described in claim 2. Li in view of Hennessy has not taught that the data transfer is rendered through specifying a range from a first specific processor element through a second specific processor element by specifying immediate values by using operands. However, Kaneko has taught such a concept. See Fig.12B and column 10, lines 14-29, and note that the global processor specifies an entire range of coprocessors by issuing a designation number in the form of P(a, b, c, d). Clearly, this will be an operand of some form of transmit (or communicate) instruction. Also, a, b, c, and d are integers, which are in turn, immediate values. A person of ordinary skill in the art would have recognized that it is advantageous to specify a range of elements in a transfer instruction so that multiple elements may receive the data at the same time, as opposed to transferring the data serially to each element one at a time. Consequently, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li to specify a range of elements.

28. Referring to claim 4, Li in view of Hennessy has taught a parallel processor as described in claim 2. Li in view of Hennessy has not explicitly taught that the data transfer is rendered through specifying bits such as to specify processor elements matching processor-element numbers expressed in binary notation and specifying processor elements by masking arbitrary bits of the thus-specified bits, through specifying the immediate values by using the operands. However, Kaneko has taught such a concept. See Fig.6 and note that a comparison occurs between the number assigned to the processor (mask) and the number specified by the global

processor. This assigned number is a mask because it is used to determine the value of bit locations in another value (the value specified by the global processor), which is the purpose of a mask. More specifically, the mask is used in a comparison. If the values are equal, then the system knows that the corresponding coprocessor (processor element) will be used in communication. And, these numbers are integers, which are in turn immediate values. A person of ordinary skill in the art would have recognized that in order to transfer data specifically intended for just one processing element, the processing element must be identified by its assigned number alone. Therefore, in order to transfer data to a single processing element, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li to include the specifics of claim 4.

29. Referring to claim 8, Li in view of Hennessy has taught a parallel processor as described in claim 7. Li in view of Hennessy has not taught that specifying of said flag bits is rendered by specifying the range from the first specific processor element through the second specific processor element through specifying the immediate values by using the operands. However, Kaneko has taught such a concept. See Fig.8A, and note that immediate values 01100 (502) and 0110 (503), which are specified by the global processor, correspond to a range of processors that will be enabled. These immediate values contain the flag bits which specify the range of elements which will be enabled. This would allow multiple processing elements to be enabled at once as opposed to enabling them one at a time. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li such that the flag bits are rendered by specifying a range of elements.

30. Referring to claim 9, Li in view of Hennessy has taught a processor as described in claim 7. Li in view of Hennessy has not taught that specifying of said flag bits is rendered through specifying bits such as to specify processor elements matching processor-element numbers expressed in binary notation and specifying processor elements by masking arbitrary bits of the thus-specified bits through specifying the immediate values by using the operands. However, Kaneko has taught such a concept. In Kaneko, it should be realized that the flag bits merely play a part in determining which processing element(s) will be enabled. Furthermore, see Fig.6 and note that a comparison occurs between the number assigned to the processor (mask) and the number specified by the global processor. This assigned number is a mask because it is used to determine the value of bit locations in another value (the value specified by the global processor), which is the purpose of a mask. More specifically, the mask is used in a comparison. If the values are equal, then the system knows that the corresponding coprocessor will be enabled. And, these numbers are integers, which are in turn immediate values. Finally, even the output of AND gate 709 in Fig.6 is a flag bit in that through specifying bits which either match or do not match the processor number, this flag is produced which either enables or disables the coprocessor from receiving input through input unit 710 for execution. A person of ordinary skill in the art would have recognized that in order to render flags specifically intended for just one processing element, the processing element must be identified by its assigned number alone. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Li to include the rendering scheme of Kaneko.

Conclusion

31. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
January 9, 2006

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100